PREDICTIVE INTERFACES FOR LONG-DISTANCE TELE-OPERATIONS

Kevin R. Wheeler[(1)], Rodney Martin[(1)], Mark B. Allan[(2)], Vytas Sunspiral[(2)(3)]

[(1)]Intelligent Systems Division, NASA Ames Research Center, MS 269-1, Moffett Field, CA, USA 94035, Email:kwheeler@mail.arc.nasa.gov
[(2)]QSS Group Inc., NASA Ames Research Center, MS 269-2, Moffett Field, CA, USA 94035
[(3)]Formerly published as Thomas Willeke

ABSTRACT

Humanoid robots have a practical advantage over other robotic platforms for use in space-based construction and maintenance because they can share tools and work interactively with astronauts. A major disadvantage is that they are difficult to control due to the large number of degrees of freedom, which makes it difficult to synthesize autonomous behavior using conventional means. We address the development of predictive tele-operator interfaces for humanoid robots with respect to two basic challenges. We first address automating the transition from fully tele-operated systems towards degrees of autonomy. We then develop compensation for the time-delay that exists when sending telemetry data from a remote operation point to robots located at low Earth orbit and beyond. Our primary goal is to show that within an operator's movement, early prediction is possible while at the same time eliminating false predictions/alarms and minimizing missed detections.

1. INTRODUCTION

The operation of humanoid robotics across great distances poses significant challenges both due to the time-delays incurred as well as to the large number of degrees of freedom. The time-lag across long distance transmissions causes "bump and wait" behavior that is far from optimal and can result in unmanageable risks. Humanoid robots such as Robonaut have a large number of degrees of freedom: two arms each with shoulder, elbow, and wrist joints, and hands with four fingers and thumb. Naturally, some of the degrees of freedom can be limited through appropriate kinematic constraints, but the number of degrees of freedom still remains high.

We are working with the NASA Johnson Space Center's Robonaut [6]: an anthropomorphic robot with fully articulated hands, arms, and neck. We have embedded trained hidden Markov models into a state machine that makes use of the command data, sensory streams, and other relevant data sources to predict a tele-operator's intent. This allows us to achieve sub-goal level commanding without the use of predefined command dictionaries. Our method works as a means to incrementally transition from manual tele-operation to semi-autonomous, supervised operation. The multi-agent laboratory experiments conducted by Bluethmann and Ambrose et. al. [6] has shown that it is feasible to directly tele-operate multiple Robonauts with humans to perform complex tasks such as truss assembly. However, once a time-delay is introduced into the system, the rate of tele-operation slows down to mimic a bump-and-wait type of activity.

We would like to maintain the same interface to the operator despite time-delays. To this end, we are developing an interface that will allow us to predict the intentions of the operator while interacting with a 3-D virtual representation of the expected state of the robot. The predictive interface anticipates the intention of the operator, and then uses this prediction to initiate appropriate sub-goal autonomy tasks.

The tele-operator's command sequence acts as the observation sequence of a hidden Markov model (HMM), in which the states of the Markov chain represent different phases of operator movement, or sub-goals. As such, any significant relative change in motion and hence intended action of the tele-operator can be related to a change in state of the hidden Markov model. We have used similar techniques in the past for the development of gesture-based computer interfaces [8].

There are several algorithms that can be used for the purposes of prediction or designing an alarm system based upon these hidden Markov models. In addition, there are several different combinations of feature vectors that may be implemented. These feature vectors act as templates for the observation sequences used to train and recall the models. These include susbsets of the pose vector, which provides position and orientation information, as well as Euclidean distances to the objects of interest being reached for.

In an effort to make our work more generally applicable to other manipulator based robotic systems, another element of our approach is to automatically decompose the physical tasks into sub-tasks in an unsupervised manner. We perform this off-line

Proc. 'ISAIRAS 2005 Conference', Munich, Germany, 5-8 September 2005 (ESA SP-603, September 2005)

analysis by Gibbs sampling a Dirichlet mixture model for isolating the key states. These states are associated with the same hidden Markov models used to predict state changes via an alarm system, corresponding to the sub-goals being targeted. Upon recognition of a sub-goal intention, a command is issued to initiate an autonomous action associated with the predicted sub-goal.

The current method of controlling Robonaut involves the operator wearing two data gloves that are used to measure finger joint positions, and two magnetic trackers used to measure the x-y-z and roll-pitch-yaw position of each hand (end effector). The position and orientation information is then transmitted to the robot as end effector position commands. The number of degrees of freedom in the elbow and shoulder are constrained to enable this position while maximizing strength. For safety considerations, the rate of movement of the arms is limited; thus the operators are trained to match or move slower than this rate. Most of the feedback to the operator comes from the stereo cameras mounted in the head of Robonaut and transmitted back to the tele-operator's head mounted display. Thus an operator will reach for an object so that his view from the head mounted cameras is not obscured by the hand. This can result in some simple tasks taking a very long time to accomplish. For example, in grasping a hand rail (as a rung) of a ladder, the operator must make sure that the fingers can wrap around the handle using the stereo visual cues. This action typically takes several seconds for an experienced operator.

One sequence of actions that we consider in this paper is reaching out for a hand rail, picking it up, and then placing it into a box. For a human to do this directly, the whole task might take 2-3 seconds. The typical time for an experienced Robonaut operator averages 15 seconds. This is the amount of time it takes when there is no time delay. When the round trip time delay is 2 seconds, this tele-operation task typically takes twice as long (due to bump-and-wait behavior).

Our approach to dealing with this time delay is to develop a predictive interface for the operator that predicts the intended action. Our goal was to keep the interface the same between full manual operations and semi-autonomous operations whereby autonomous commands (e.g. grasping) are issued. It might have been easier to create a system by which the autonomous operations are commanded by making a list of them, or by touching symbols on the screen. But two issues arise: scalability and adjustable autonomy.

Scalability:
Since Robonaut is a humanoid robot, the potential number of movements (or sub-goals) that it is able to accomplish is very large for typical maintenance and assembly tasks. This means that as the numbers of tasks increases and as the complexity of the tasks increases the number of commands increases (potentially exponentially). This makes an abstract symbolic interface potentially cumbersome.

Adjustable Autonomy:
In spite of the level of sophistication of the autonomy onboard Robonaut, it will still be necessary to allow for human intervention. One might imagine complex and unexpected emergency maintenance tasks where only some of the autonomous behaviors necessary to complete the task are available. In an emergency we cannot wait for these autonomous behaviors to be developed. Rather it will become necessary to have fully manual tele-operation to fill these critical gaps. Thus we expect that there will always be the need to have the ability to seamlessly transition from autonomous operations to manual control.

The interface that we are developing sends the prediction of the tele-operator's intent over a time delay to a Robonaut that has autonomous behaviors such as grasping. Our current approach consists of building a finite state machine at a high level of abstraction containing embedded hidden Markov models at a hierarchically lower level. To build these autonomous models it is necessary to identify the appropriate state/task decomposition. We automatically decompose the task into states using mixtures of Dirichlet distributions to model each behavior with the HMMs embedded within the state machine. This decomposition was used to validate the numbers of states and mixtures selected heuristically by an empirical analysis of the observed data

In the following sections of this paper we describe the experiments conducted, the methodology followed, and the results. The experiments section describes the different experiments performed. The methodology section describes the data available from Robonaut, the Dirichlet based task decomposition, the hierarchical models employed, the algorithms used for prediction and alarm, and the testing and validation procedures. The results section discusses performance issues as well as alternative approaches.

2. EXPERIMENTS

We chose two basic tasks, retrieving a hand rail mounted vertically and dropping it into a box, and retrieving a hand rail mounted horizontally and dropping it into a box. The hand rails are mounted

with Velcro on a cloth board, affixed to a stationary wall. The target box is a flexible cloth box that is open but is not within the same field of view as the hand rails. These tasks were chosen as a first step towards automating climbing on a space habitat.

The tasks consist of the following steps:

1. start in initial position/state
2. look down at hand (substitute for proprio-receptive feedback) and then at hand rails
3. reach for specified hand rail (either vertical or horizontal according to plan)
4. grasp hand rail
5. remove hand rail from wall (pull)
6. move hand rail over box
7. drop hand rail into box
8. return to initial position

The Robonaut can be operated via a simulated environment, so that the operators can perform tasks without regard for the time-delay normally associated with long distance operations. For this experiment, inexperienced operators tended to have greatly varying behaviors, whereas the variance in the data was negligible for the most experienced tele-operator.

Fig. 1 shows the simulated environment in which the experiments discussed in this paper were conducted. These experiments were conducted on six different days spanning over three months. Comparisons are made between different operators and the same operator on different days. Initial conditions varied noticeably from day to day.

3. METHODOLOGY

The data used in our study is the command data coming from the operator. This data consists of the desired position and orientation of the end effector (right hand) as well as the joint angles of the fingers and thumb of the right hand. Position and orientation data is useful in forming observation sequences with which to train hidden Markov models. In this article, we only focus on one-handed tasks. Output information is also available from the Robonaut, but we do not use this information because we wish not to wait for data as it crosses the time-delay.

The Robonaut also makes use of a Sensory Ego Sphere (SES) [5] that stores object identification and position information. The SES serves as a short-term memory structure for the robot, and much of the information stored in the SES is obtained from the machine vision object recognition system. By monitoring updates to the SES, we are able to detect when new objects enter the robot's field of view. The distances between an end effector and the objects in the robot's sphere of influence can be used as an additional sensory stream. This is also useful in constructing feature vectors with which to train HMMs.

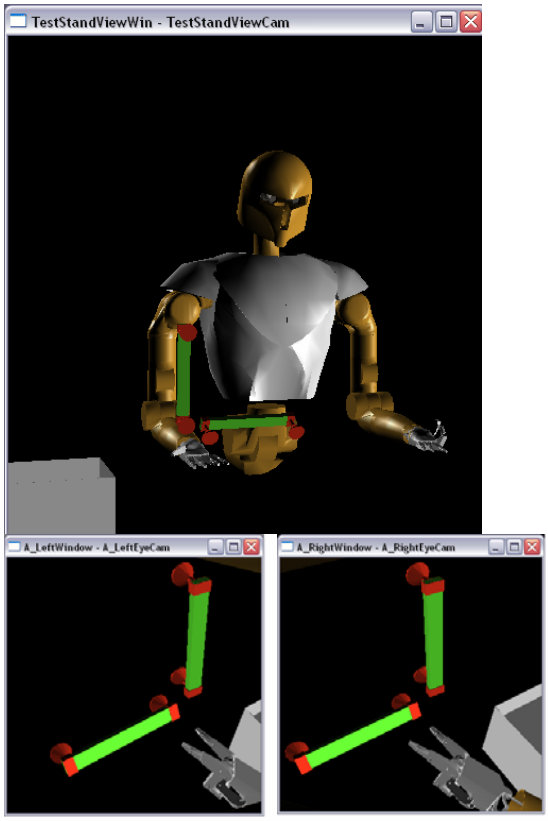

Fig. 1. Actual Simulation based experiment. The operator view from the left and right cameras of the simulated hand rails.

We have developed a state machine which models both tasks of reaching for the horizontal or vertical hand rails by embedding Hidden Markov models (HMMs) at each state within the state machine, shown in Fig. 2. Thus the movement corresponding to reaching for both the vertical and the horizontal hand rails are each modelled with a distinct HMM. The hidden states within each HMM are determined automatically through the use of Dirichlet process mixtures, off-line. These states may not necessarily correspond to a human semantic notion. The whole task can be decomposed using this same methodology so that we have multiple layers of hidden states. In the work described here, the higher-level states are explicitly described to correspond to the given tasks so that it can be mapped to the planning system.

In this application, we have decided that it is more desirable to have no false predictions (alarms) with the possibility of having missed detections. This decision was made because the tele-operator will remain in

manual operation mode through out the operation if no prediction occurs. If a prediction occurs, then this prediction will be used to trigger the autonomous grasping behavior at the robot (upon operator concurrence).

3.1 Movement modelling

A simplified version of the state-machine employed in this work is shown in Fig. 2. This relates directly to the 8-step task decomposition described at the beginning of the previous section. State M0 (Start) refers to Step #1: the starting/initial state. The two "Reach" states, where the state machine bifurcates, correspond to Step #3: reach for specified hand rail (either vertical or horizontal according to plan). State M3 (Grasp) refers to Step #4, grasp hand rail, State M4 (Move to Box), corresponds to Step #6, move hand rail over box, and State M5 (Drop) refers to Step #7, drop hand rail into box.

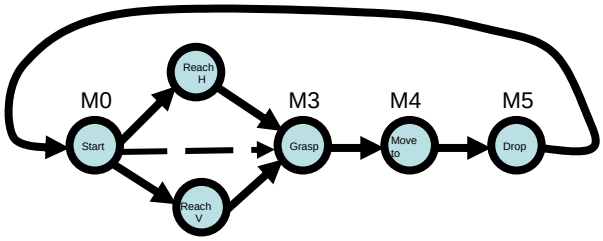

Fig. 2 State machine with embedded HMMs.

The HMMs are trained on segmented operator data consisting of subsets of x-y-z and roll-pitch-yaw (pose vector) information. The HMMs are parameterized using left-to-right tied mixtures Baum-Welch training. The recall is performed with Viterbi recall in order to monitor the best state sequences, and to record the log of the likelihood of the optimal sequence to serve as both a condensed metric, and as a method for arbitration between competing models. [1-3]. Alternatives to this recall method are also available and will be discussed in a subsequent subsection. The initialization of the HMMs and the state machine are done using the posterior from a Dirichlet process [4].

3.2 Dirichlet processes for task decomposition

To automate the task decomposition into states, where the continuous observation variables associated with each state are modelled as mixtures of Gaussians, we have used a Dirichlet mixture process [4]. This works by assuming that each observation is modelled using a normal distribution with unknown mean and variance:

$$y_i \sim N(\mu_i, \tau) \tag{1}$$

The means are modelled as coming from normal distribution conditioned upon an unknown class which comes from a Dirichlet process:

$$\mu_i = \lambda_{P_i} \tag{2}$$

$$\lambda \sim N(0, \varepsilon) \tag{3}$$

$$P_i \sim D(\alpha) \tag{4}$$

We then use a Gibbs sampler to determine the proper parameters for the given data. The resulting posterior $p(\mu \mid y)$ is then used to determine the number and locations of states for each task. Note that this says nothing about the shapes of the distributions of the means; this only provides an understanding of how many states may be required. Ideally, this would be incorporated inside of an iterative system whereby we simultaneously and hierarchically model the shape as well. One of the resulting posteriors is shown in Fig. 3 for the *y* commanded position.

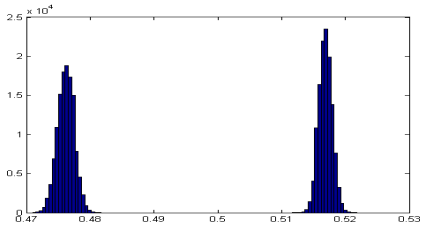

Fig. 3. Gibbs sampled posterior for means of states. .

The number of peaks apparent in Fig. 3 represents the desired number of states to decompose the task into for a single dimension. In this figure, two states were identified, and multiple dimensions were then combined.

We used OpenBUGS [7] to parameterize the Dirichlet process from the observation streams. Due to difficulties with initialization of the Markov chains within OpenBUGS for our multi-dimensional model, we decided to model each stream independently. Thus the independent estimates of the number of states for each dimension had to be combined when training the HMMs. In our system we used three states with six mixtures for modelling the operator command data stream.

3.3 Voting scheme

The HMMs were trained using data that was segmented and aligned. In real-time operations, the data is streaming into the system and is not segmented or aligned. For task prediction, we implement a

combination of a probability detection threshold with a voting scheme. The detection threshold allows for arbitration between competing recalled models**.**

The voting scheme arbitrates between the two data streams for an event to be detected by choosing the model that has the highest likelihood several consecutive times as well as a large confidence. This is done primarily to prevent false detections. We define confidence in terms of the difference in log probabilities between the models (for more than two models this is the difference between the two highest likelihood models). This confidence value is then transmitted to the Sensory Ego Sphere in order to update the color encoding of the hand rails. Initially the hand rails start off as green in appearance. As the confidence of the predictor increases through out the movement, the color of the predicted hand rail changes from green to red.

3.4 Alarm systems

One type of alarm system is based upon the output of an HMM in order to predict the action of a tele-operator as early possible into an operator's movement, while at the same time eliminating false alarms and minimizing missed detections. This alarm system essentially consumes the information available by any real-time processing that occurs as a result of using HMMs. An example of such an alarm system is the prediction method described in the previous sub-section. In this case, the detection threshold, voting scheme, and confidence thresholds all serve as alarm system design parameters.

Ultimately, we would like to be able to implement an *optimal* alarm system which can be designed apriori that optimizes the metrics we've described. However, as a first step, in this paper we will only compare alarm-based statistics for prediction/alarm algorithms that are designed or devised heuristically. These alarm –based statistics include the aforementioned probabilities of false alarm and missed detection, as well as the time to prediction. By implementing different algorithms which are fundamentally and theoretically sub-optimal with respect to the defined metrics, we should be able to determine the most suitable alarm system from an implementation standpoint by directly comparing the metrics and requirements, given that both were trained on the same data set.

Consequently, as an alternative to the voting method described in the previous section, we may also use real-time recall based upon the posterior probabilities of the HMMs rather than the Viterbi recall method. This method returns an alarm if the maximum probability of the state that we occupy within the hidden Markov model over the span of the data block under consideration exceeds a certain confidence threshold.

A condition on the state we occupy also exists hinging upon the fact that the HMMs are trained as left-right models. We expect that the state sequence will proceed from left to right, meaning that any change in state should occur in only one direction. In this case, any change from the initial state to a subsequent state of one model while the other model is still in the initial state indicates an operator reaching action with a confidence level given by the posterior probability described previously. This is the only other alarm-based method that we'll examine in this paper out of many potential candidate alarm-based methods.

4. RESULTS

Two types of results are of interest to us in this work. The first is the most common in the machine learning literature and consists of prediction error results on validation sets. The second type of result consists of performance during operation both in terms of the prediction time as well as false alarms/missed detections analysis.

4.1 Batch file validation

4.1.1 Single operator models

We had a number of different features from which to select for training the HMMs within the state machine. We focused upon using only variables which directly came as commands from the tele-operator. These variables include the x-y-z position and roll-pitch-yaw orientation of the end effector, and the joint angles of each of the fingers. Finger joint angles are used to determine hand open and hand closed states. To select which variables to use for modelling we used the following criteria:

1. low variability between operators
2. low variability with respect to initial conditions
3. large Kullback-Leibler (KL) distances between variables for different tasks (grasping horizontal rail vs. grasping vertical rail)

Initially we had intended to use hand pre-shape as an indicator of intention with respect to which object was to be grasped. However, we decided not to pursue this approach for the experiments described. The current experimental setup calls for two identical handrails that differ in position and orientation, but not in shape. More importantly, we observed that the operators tend

to pre-flatten their hands while reaching for most objects regardless of shape.

Given the considerations between minimizing operator variance and maximizing the KL distance, we elected to use *yaw* as the primary variable indicative of orientation. The best positional variables were *y* (which represents side to side motion) and *z* (which represents up and down motion).

This feature vector may also be replaced with or augmented with the Euclidean distance to both the vertical and the horizontal hand rails as determined by the machine vision system. The advantage of using this feature is that it provides more discriminatory power that is based more closely on the experimental setup rather than the operator's pose.

It is important to note that these feature vectors only have bearing on the model, not on the method of prediction (i.e. which type of alarm system is used). As a result, it can be surmised that the alarm system itself, including its design and implementation, is independent of the parameters and training regimen of the hidden Markov model. This need not necessarily be the case, and in fact it may be possible with further theoretical investigation to integrate the two. However, for now these candidate feature vectors can be used either in conjunction with the voting scheme based upon the Viterbi recall method or the alternate one based upon posterior probabilities described in the previous section.

To test our training methods we have trained an HMM on reaching for the horizontal hand rail and another HMM on a vertical hand rail on 36 trials collected over three months. These trained models were then tested on 20 different trials, using the Viterbi recall method and the *y/yaw* feature vector. All of these trials were for a single operator. The HMMs were trained using left-to-right transition matrices with tied Gaussian mixture models consisting of three states and six mixtures. The recall on the unseen validation set resulted in 100% recognition with no errors. The sampling rate of the data used in training was 15 Hz.

4.1.2 Multiple operator models

The development of an operator independent model requires a considerable amount of data. Nonetheless, we were able to analyze some differences between operators as well as to train models across operators and test them. Fig. 4 shows the path of the commanded end effector for a single dimension for two operators (operators # 1 and 3) collected on different days. Fig. 5 shows the same path for operator #1 only, across two different days. Fig. 6 shows the same path for operator #1 over three months encompassing five different data collections. Note that the vertical path has bimodal behavior when spanning this many different trials.

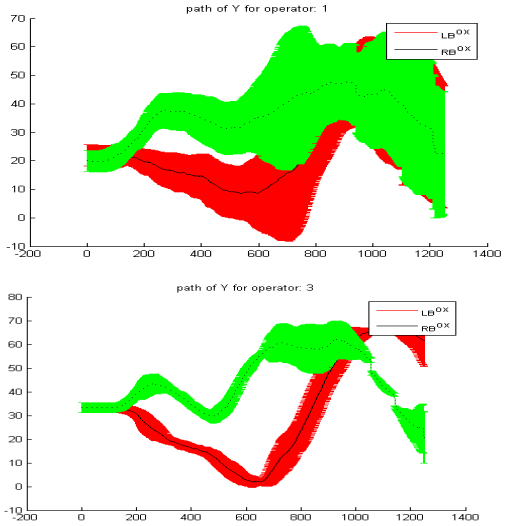

Fig. 4. Trajectory with error bars for *y* for reaching for horizontal and vertical (red & green). An inexperienced operator is shown top, experienced operator on bottom with minimal variance across 6 trials.

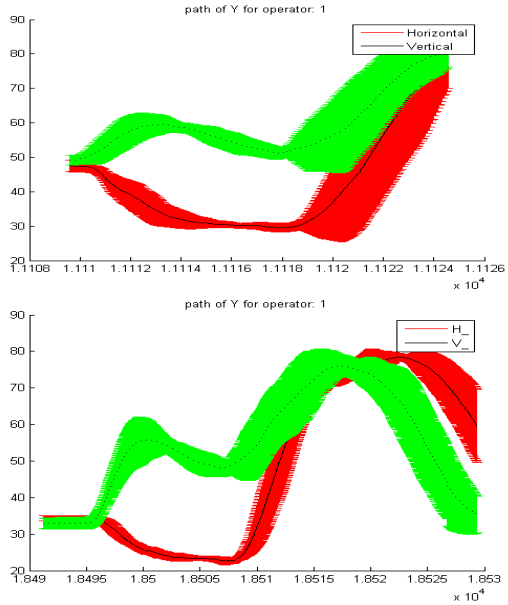

Fig. 5 Trajectory of *y* variable for an experienced operator for two different days. Red is reaching for horizontal, green is reaching for vertical. Horizontal axis is time in seconds, vertical is reach in cm.

One effect that we observed is that when an operator has not had enough experience operating the actual hardware, their performance changes within the simulated environment in a manner not consistent with operation of the hardware. Another effect is that the operators tend to modify their behaviors over time as they learn how to optimize their performance with respect to time and force minimization. This modification over time causes a non-stationarity to occur that we currently do not model. To build the models presented in this paper we used operators that had an experience level that has fully converged onto stationary patterns.

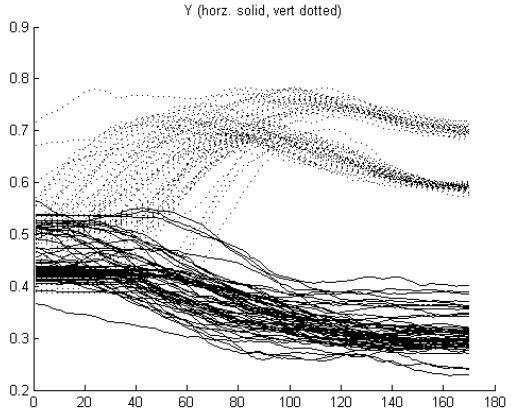

Fig. 6 The trajectory of variable *y* for the same operator spanning 3 months incorporating 5 data collection periods. Solid lines are for reaching for horizontal hand rail, dotted lines are for reaching for vertical. Horizontal axis is in samples, vertical axis is in normalized (0, 1) units.

An alternative approach is to have a *calibration* period at the beginning of each day of operation that will allow for us to adapt to the non-stationarity by modifying the models built previously with the current day's actions. The draw-back to this is that it requires the operators to endure yet more preparation and does not prevent the operators from behaving differently during the actual operation.

4.2 Real-time performance

We study the real-time performance of our system by measuring the length of prediction before the operator grasps the object of interest. We determine the grasp by looking at the average base pitch of the fingers of the operator's Cyberglove. Our goal was to be able to predict at least two seconds before the grasp. For the Viterbi recall method, our average prediction time was five seconds before grasp with zero false alarms and one missed detection out of the 30 trials in the validation set. We could increase the prediction time before the grasp by accepting more false alarms. However, in a tele-operation environment it was decided that it is far better to miss detections than to falsely try to initiate an inappropriate autonomous behaviour. The prediction time could be increase in a variety of ways, including changing the alarm system parameters (thresholds with Viterbi), the alarm algorithm itself (using posterior rather than likelihood), as well as the feature vector.

This can clearly be demonstrated by the real-time recall results for the different methods and feature vectors. Table 1 illustrates the metrics of interest for using either the Viterbi or the posterior recall methods, with feature vectors based on either y/yaw, or distances. In this table, the "avg. time" column refers to the average time before the grasp in seconds, for correct predictions only. The training and test sets used were the same as for the static recall results.

Recall Method	Feature Vector	False Alarms	Missed Detections	Avg. Time
Posterior	y/yaw	15	0	11.44
Viterbi	y/yaw	0	1	6.4
Posterior	Distance	7	7	7.5
Viterbi	Distance	0	0	4.1

Table 1. Dynamic Recall Results based on a single session. Average prediction time before grasp is in seconds, larger number is better.

As seen in Table 1, there is a trade-off between the alarm statistics (false alarms and missed detections), and the average prediction time prior to the grasp. This tradeoff is related to the recall algorithm used (i.e. the posterior vs. the Viterbi recall method) and the thresholds employed. In our operator interface we have chosen to eliminate false alarms at the expense of decreasing the length of prediction ahead of grasping. Thus the thresholds used with the Viterbi method have been established to minimize false alarms. It would have also been possible to set the thresholds associated with the Viterbi method so as to increase the prediction window length but this has the affect of non-zero false alarms.

The tele-operator working within the simulated environment becomes aware of the intent prediction via a change in color of the predicted hand rail in the software interface. If this is the correct hand rail, the operator merely needs to close his hand and the autonomous grasping action will be enacted if the predictors confidence is high enough. If the prediction is wrong, the operator can ignore the color changes and continue to perform in full manual operation.

The level of confidence in the prediction is transmitted as a floating point value (between zero and one) to one of the console officers monitoring the performance of the system. If low values of confidence are accepted, then it becomes possible to have very early predictions at the cost of having false alarms. In this study, we have elected to minimize false alarms and therefore our average prediction time remains at approximately six seconds prior to the grasp. However, to alleviate the need to have an *optimal* threshold, we encode the confidence information into the colors of the hand rails. Thus the tele-operator can see the transition of the prediction confidence as the gestural command progresses through time.

5. CONCULSION

The development of a predictive interface for tele-operators has been discussed. We have introduced the concept of detecting the peaks in the posterior of Dirichlet based model for automating task decomposition. In addition, we have built and trained HMMs embedded within a state machine as a means to predict the intentions of tele-operators. Our average prediction ahead of task completion of 6 seconds allows for us to compensate for our target time of 2 seconds round trip time delay. We have demonstrated the trade-off between low false alarms and missed detections versus desirable prediction times. We have also tested this system within a live tele-operation environment at NASA JSC with successful runs of zero false alarms and zero missed detections.

6. ACKNOWLEDGEMENTS

The authors wish to thank the tele-operators Mike (the guru), Josh, Nick and Darby for their time and patience. We would also like to thank Kim Hambuchen for helping us with the SES and network code. We are especially grateful for Bill Bluethmann's consideration and advice and Rob Ambrose's vision in making this project possible at NASA JSC.

7. REFERENCES

1. Rabiner L. R., "A Tutorial on Hidden Markov Models and Selected Applications in speech Recognition," *Proc. IEEE*, vol. 77, no. 2, Feb., 1989, pp.257-287.
2. Bellegarda J. R. and Nahamoo D., "Tied Mixture Continuous Parameter Modeling for Speech Recognition," *IEEE Trans. Acoustics, speech, and Signal Processing*, vol. 38, no. 12, Dec., 1990, pp. 2003-2045.
3. Forney Jr. G. D., "The Viterbi Algorithm," *Proc. IEEE*, vol. 61, no. 3, Mar. 1973, pp. 268-278.
4. Escobar M. D. and West M., "Computing Nonparametric Hierarchical Models," *Practical Nonparametric and Semiparametric Bayesian Statistics*, Eds. Dipak Dey, Peter Muller, Debajyoti Sinha, Springer-Verlag, NY, pp.1-22, 1998.
5. Peters II R. A., Hambuchen K. E., Kawamura K., and Wilkes D. M., "The Sensory Ego-Sphere as a short-Term Memory for Humanoids", *Proc. IEEE-RAS Int'l. Conf. on Humanoid Robots*, Waseda University, Tokyo, Japan, Nov. 22-24 Nov., pp. 451-459.
6. Bluethmann, W., Ambrose, R., Diftler, M., Askew, S., Huber, E., Goza, M., Rehnmark, F., Lovchik, C., Magruder, D., "Robonaut: A Robot Designed to Work with Humans in Space," *Autonomous Robots*, n. 2/3, v.14, 2003, pp 179-198.
7. Spiegelhalter D., Thomas A., Best N. and Lunn D., *WinBUGS User Manual Version 2.0*, June 2004, http://www.mrc-bsu.cam.ac.uk/bugs.
8. Wheeler K., Jorgensen C., "Gestures as Input: Neuroelectric Joysticks and Keyboards", *IEEE Pervasive Computing*, Vol. 2, No. 2, April-June, 2003.